# Diabetes_Prediction_Model

March 23, 2024

[49]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, confusion_matrix, recall_score,␣
 ↪f1_score, precision_score
from sklearn.linear_model import LogisticRegression
from sklearn.neighbors import KNeighborsClassifier
from sklearn.ensemble import GradientBoostingClassifier


%matplotlib inline

import warnings
warnings.filterwarnings('ignore')
```

[2]:
```
df = pd.read_csv('diabetes.csv')
```

[3]:
```
df.head()
```

[3]:
```
   Pregnancies  Glucose  BloodPressure  SkinThickness  Insulin   BMI  \
0            6      148             72             35        0  33.6
1            1       85             66             29        0  26.6
2            8      183             64              0        0  23.3
3            1       89             66             23       94  28.1
4            0      137             40             35      168  43.1

   DiabetesPedigreeFunction  Age  Outcome
0                     0.627   50        1
1                     0.351   31        0
2                     0.672   32        1
3                     0.167   21        0
4                     2.288   33        1
```

[4]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 768 entries, 0 to 767
Data columns (total 9 columns):
 #   Column                    Non-Null Count  Dtype
---  ------                    --------------  -----
 0   Pregnancies               768 non-null    int64
 1   Glucose                   768 non-null    int64
 2   BloodPressure             768 non-null    int64
 3   SkinThickness             768 non-null    int64
 4   Insulin                   768 non-null    int64
 5   BMI                       768 non-null    float64
 6   DiabetesPedigreeFunction  768 non-null    float64
 7   Age                       768 non-null    int64
 8   Outcome                   768 non-null    int64
dtypes: float64(2), int64(7)
memory usage: 54.1 KB
```

[5]:
```
df['Outcome'].value_counts()
```

[5]:
```
0    500
1    268
Name: Outcome, dtype: int64
```

[6]:
```
df.describe()
```

[6]:
```
       Pregnancies     Glucose  BloodPressure  SkinThickness     Insulin  \
count   768.000000  768.000000     768.000000     768.000000  768.000000
mean      3.845052  120.894531      69.105469      20.536458   79.799479
std       3.369578   31.972618      19.355807      15.952218  115.244002
min       0.000000    0.000000       0.000000       0.000000    0.000000
25%       1.000000   99.000000      62.000000       0.000000    0.000000
50%       3.000000  117.000000      72.000000      23.000000   30.500000
75%       6.000000  140.250000      80.000000      32.000000  127.250000
max      17.000000  199.000000     122.000000      99.000000  846.000000

              BMI  DiabetesPedigreeFunction         Age     Outcome
count  768.000000                768.000000  768.000000  768.000000
mean    31.992578                  0.471876   33.240885    0.348958
std      7.884160                  0.331329   11.760232    0.476951
min      0.000000                  0.078000   21.000000    0.000000
25%     27.300000                  0.243750   24.000000    0.000000
50%     32.000000                  0.372500   29.000000    0.000000
75%     36.600000                  0.626250   41.000000    1.000000
max     67.100000                  2.420000   81.000000    1.000000
```

[7]:
```
df.isnull().sum()
```

[7]:
```
Pregnancies                 0
Glucose                     0
BloodPressure               0
SkinThickness               0
Insulin                     0
BMI                         0
DiabetesPedigreeFunction    0
Age                         0
Outcome                     0
dtype: int64
```

[8]:
```
list(df.columns)
```

[8]:
```
['Pregnancies',
 'Glucose',
 'BloodPressure',
 'SkinThickness',
 'Insulin',
 'BMI',
 'DiabetesPedigreeFunction',
 'Age',
 'Outcome']
```

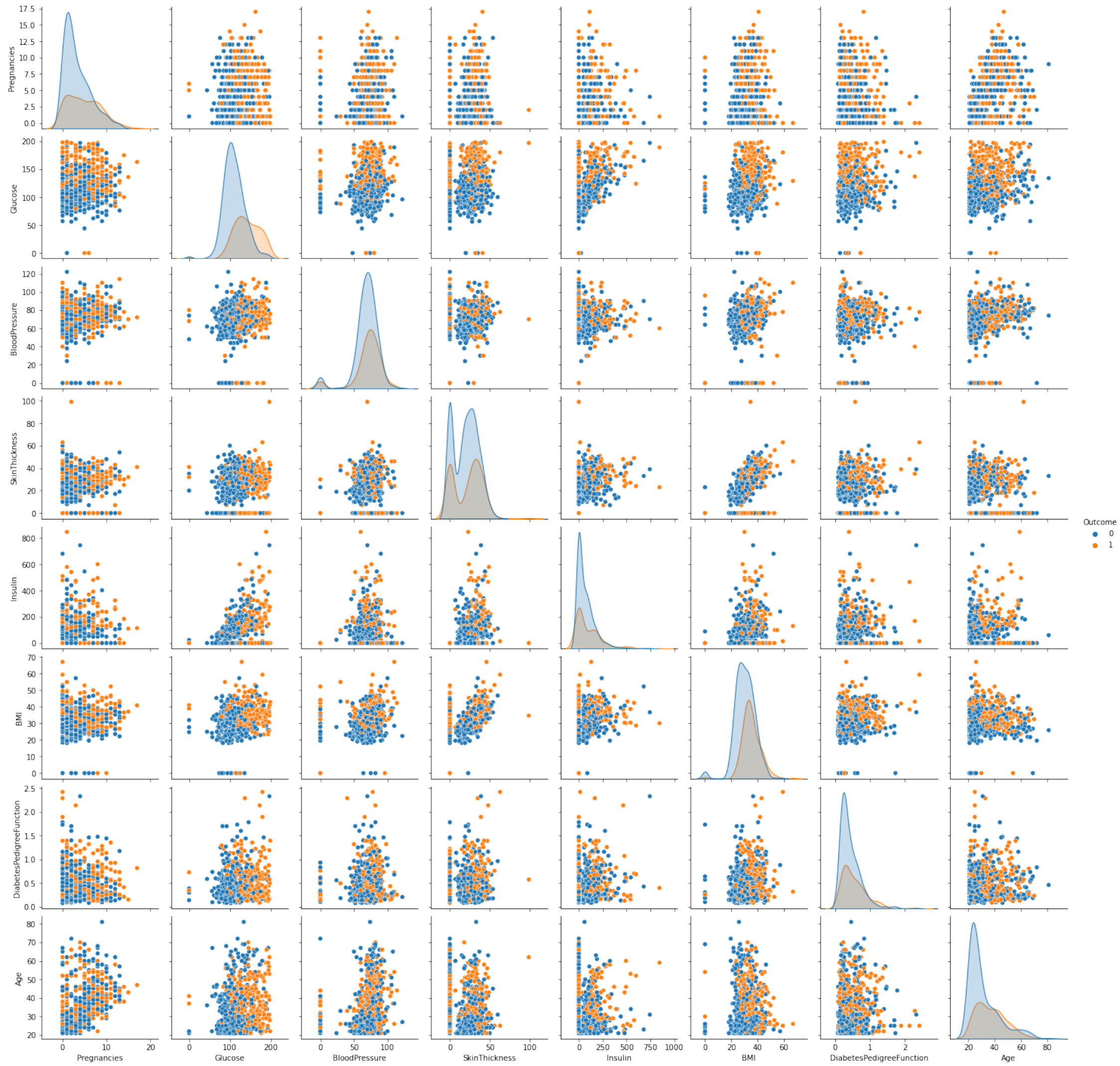
[9]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 768 entries, 0 to 767
Data columns (total 9 columns):
 #   Column                    Non-Null Count  Dtype
---  ------                    --------------  -----
 0   Pregnancies               768 non-null    int64
 1   Glucose                   768 non-null    int64
 2   BloodPressure             768 non-null    int64
 3   SkinThickness             768 non-null    int64
 4   Insulin                   768 non-null    int64
 5   BMI                       768 non-null    float64
 6   DiabetesPedigreeFunction  768 non-null    float64
 7   Age                       768 non-null    int64
 8   Outcome                   768 non-null    int64
dtypes: float64(2), int64(7)
memory usage: 54.1 KB
```

## 0.1 Exploratory Data Analysis

[10]:
```
sns.pairplot(df, hue='Outcome')
```

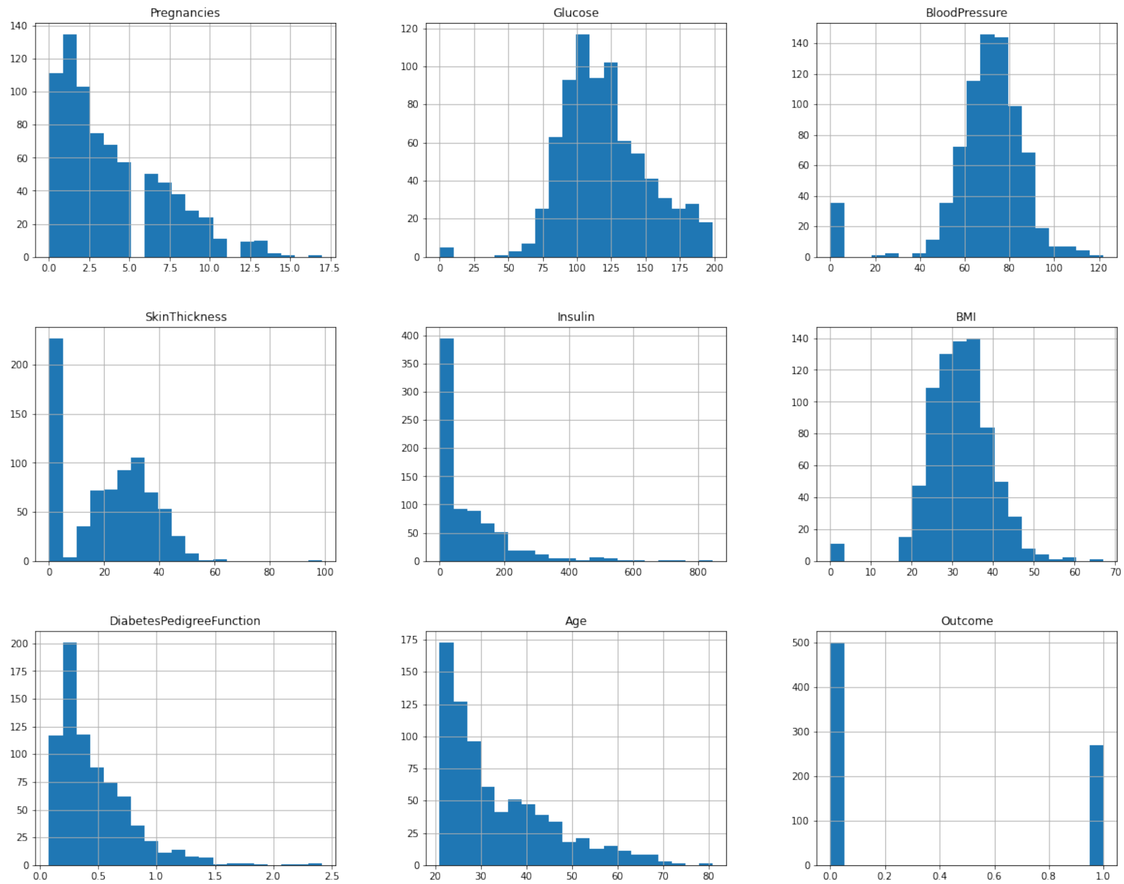
[10]:
```
<seaborn.axisgrid.PairGrid at 0x179cf8a4670>
```



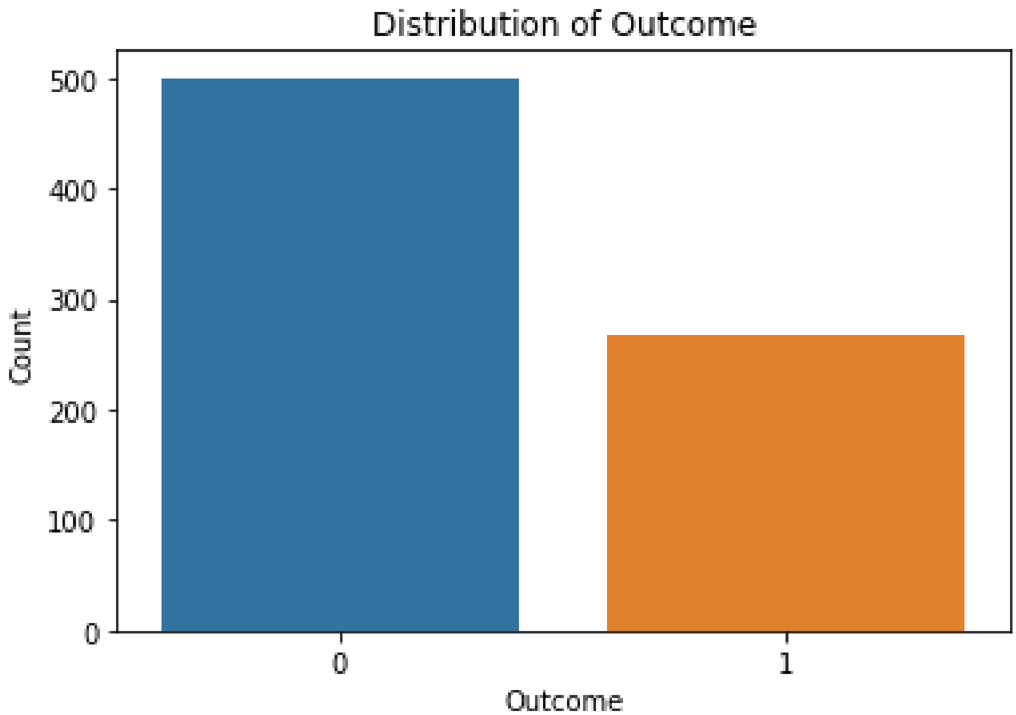

[11]:

```
df.hist(bins = 20, figsize = (20,16))

plt.show()
```

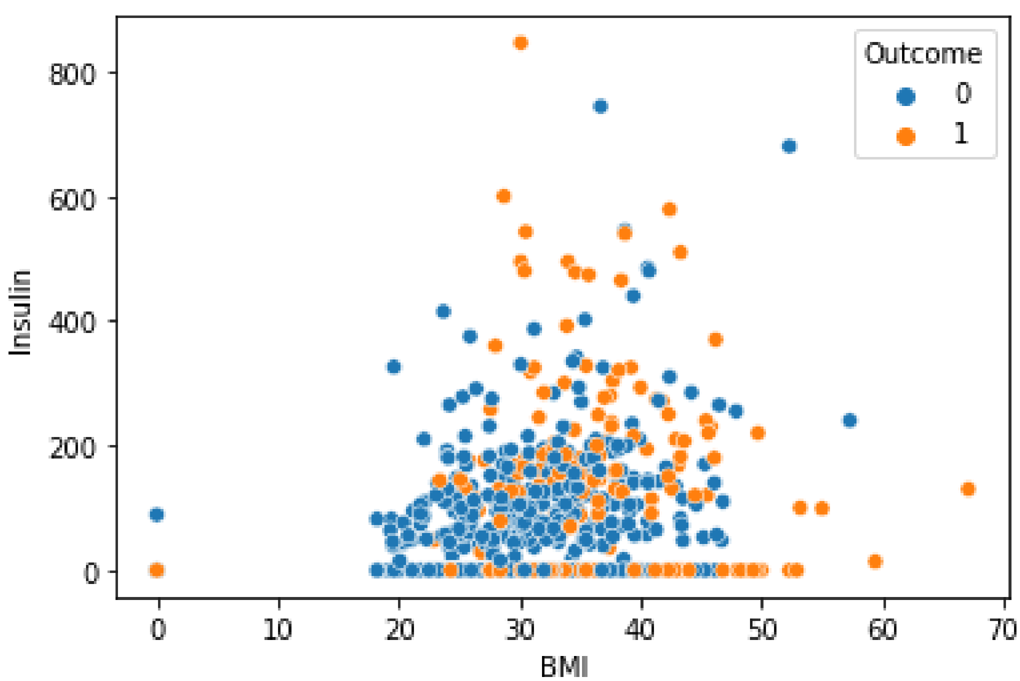



[12]:
```
sns.countplot(x = 'Outcome', data = df)
plt.xlabel('Outcome')
plt.ylabel('Count')
plt.title('Distribution of Outcome')
plt.show()
```




[13]:
```
sns.scatterplot(x = 'BMI', y = 'Insulin', data = df, hue = 'Outcome')
```

[13]: <AxesSubplot:xlabel='BMI', ylabel='Insulin'>



### 0.1.1 More on Body Mass Index

**BMI is a measure that relates body weight to height. BMI is sometimes used to measure total body fat and whether a person is a healthy weight.**

According to https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4457375/, Having even moderately elevated BMI is associated with increased risk of developing Diabetes Mellitus complications.

For this analysis the BMI will be categorized as follows: 1. Underweight: BMI less than 18.5 2. Normal weight: BMI between 18.5 and 24.9 3. Overweight: BMI between 25 and 29.9 4. Obesity: BMI of 30 or higher

[14]:
```
# Creating a function to categorize BMI
def bmi_category(bmi):
    if bmi < 18.5:
        return 'Underweight'
    elif 18.5 <= bmi  <= 24.9:
        return 'Normal weight'
    elif 25 <= bmi < 29.9:
        return 'Overweight'
    else:
        return 'Obese'
```

[15]:
```
#Applying the Function
df['BMI_categories'] = df['BMI'].apply(bmi_category)
```

[16]:
```
df.head()
```

[16]:
```
   Pregnancies  Glucose  BloodPressure  SkinThickness  Insulin   BMI  \
0            6      148             72             35        0  33.6
1            1       85             66             29        0  26.6
2            8      183             64              0        0  23.3
3            1       89             66             23       94  28.1
4            0      137             40             35      168  43.1

   DiabetesPedigreeFunction  Age  Outcome BMI_categories
0                     0.627   50        1          Obese
1                     0.351   31        0     Overweight
2                     0.672   32        1  Normal weight
3                     0.167   21        0     Overweight
4                     2.288   33        1          Obese
```

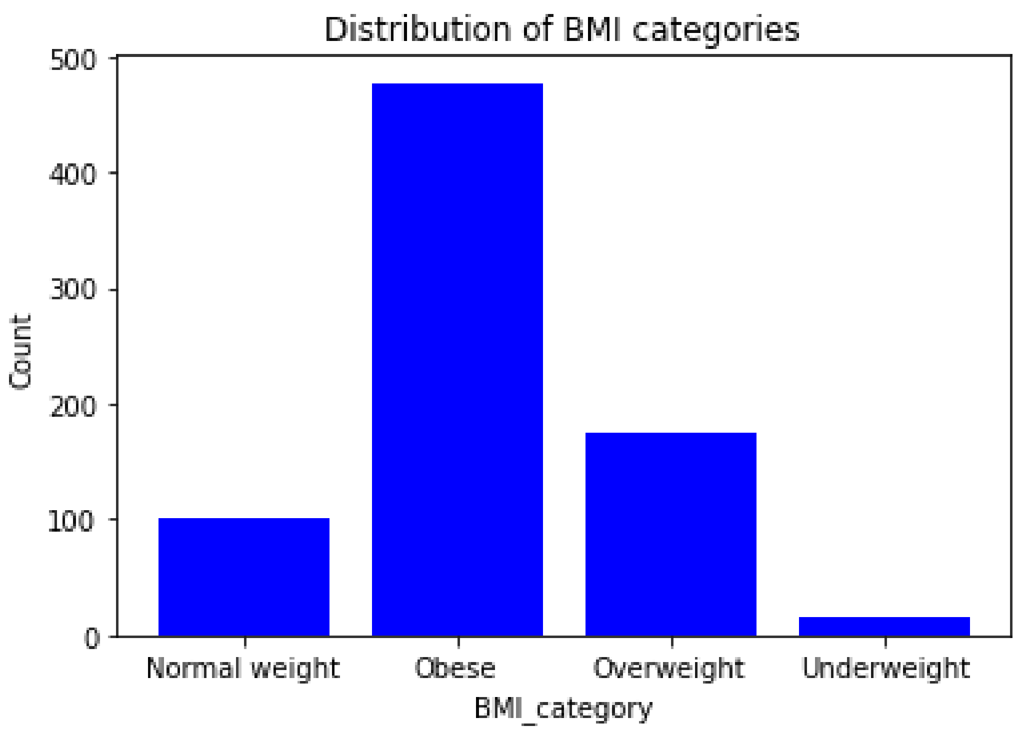
[17]:
```
df['BMI_categories'].value_counts()
```

[17]:
```
Obese            477
Overweight       174
```

```
Normal weight    102
Underweight       15
Name: BMI_categories, dtype: int64
```

[18]:
```
BMI_category_list = ['Normal weight', 'Obese', 'Overweight', 'Underweight']
```

[19]:
```
plt.bar(BMI_category_list, df.groupby('BMI_categories')['Outcome'].count(),␣
 ↪color = 'b')
plt.xlabel('BMI_category')
plt.ylabel('Count')
plt.title('Distribution of BMI categories')
plt.show()
```




[20]:
```
# Extracting data with diabetic individuals
diabetic_patients = df[df['Outcome'] == 1]
```

[21]:
```
diabetic_patients.head(10)
```

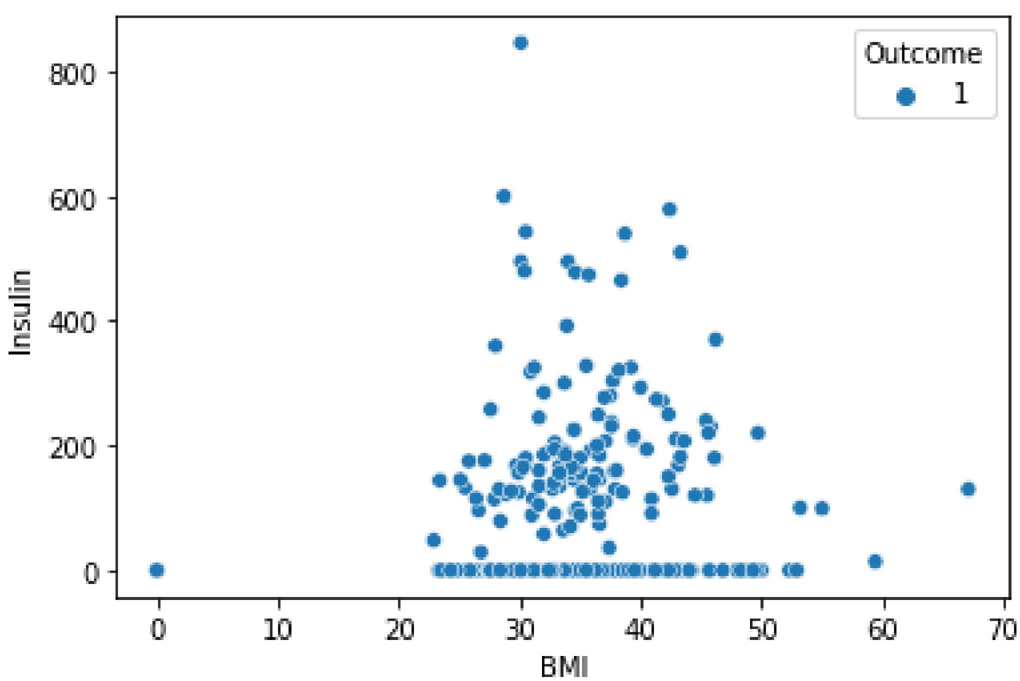
[21]:
```
   Pregnancies  Glucose  BloodPressure  SkinThickness  Insulin   BMI  \
0            6      148             72             35        0  33.6
2            8      183             64              0        0  23.3
4            0      137             40             35      168  43.1
6            3       78             50             32       88  31.0
```

```
8             2    197            70             45      543  30.5
9             8    125            96              0        0   0.0
11           10    168            74              0        0  38.0
13            1    189            60             23      846  30.1
14            5    166            72             19      175  25.8
15            7    100             0              0        0  30.0

    DiabetesPedigreeFunction  Age  Outcome BMI_categories
0                      0.627   50        1          Obese
2                      0.672   32        1  Normal weight
4                      2.288   33        1          Obese
6                      0.248   26        1          Obese
8                      0.158   53        1          Obese
9                      0.232   54        1    Underweight
11                     0.537   34        1          Obese
13                     0.398   59        1          Obese
14                     0.587   51        1     Overweight
15                     0.484   32        1          Obese
```

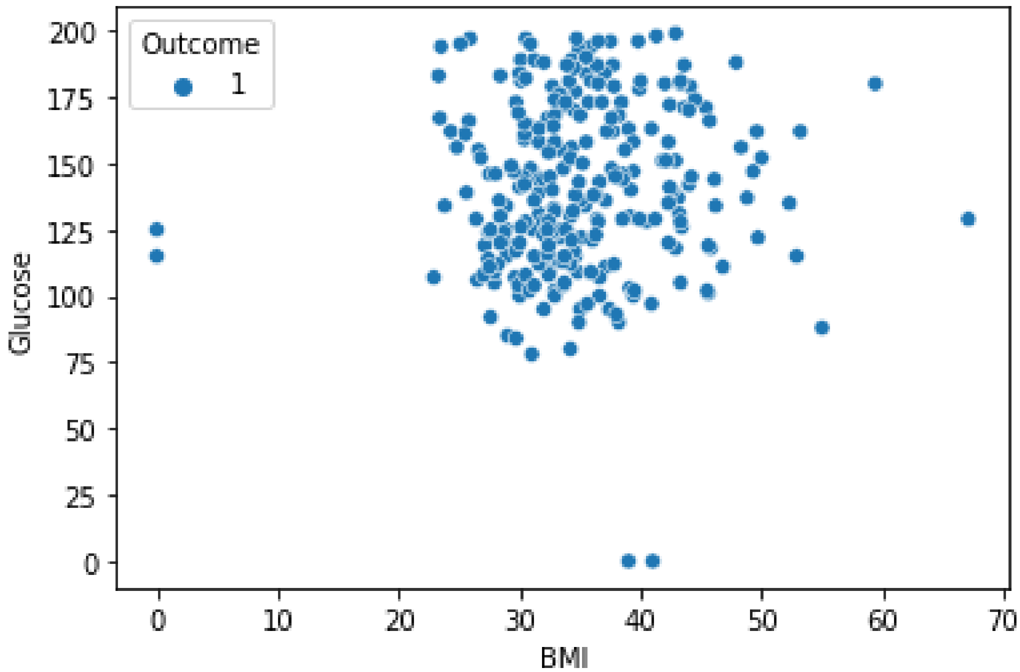
[22]:
```python
sns.scatterplot(x = 'BMI', y = 'Insulin', data = diabetic_patients, hue =␣
 ↪'Outcome')
```

[22]:
```
<AxesSubplot:xlabel='BMI', ylabel='Insulin'>
```



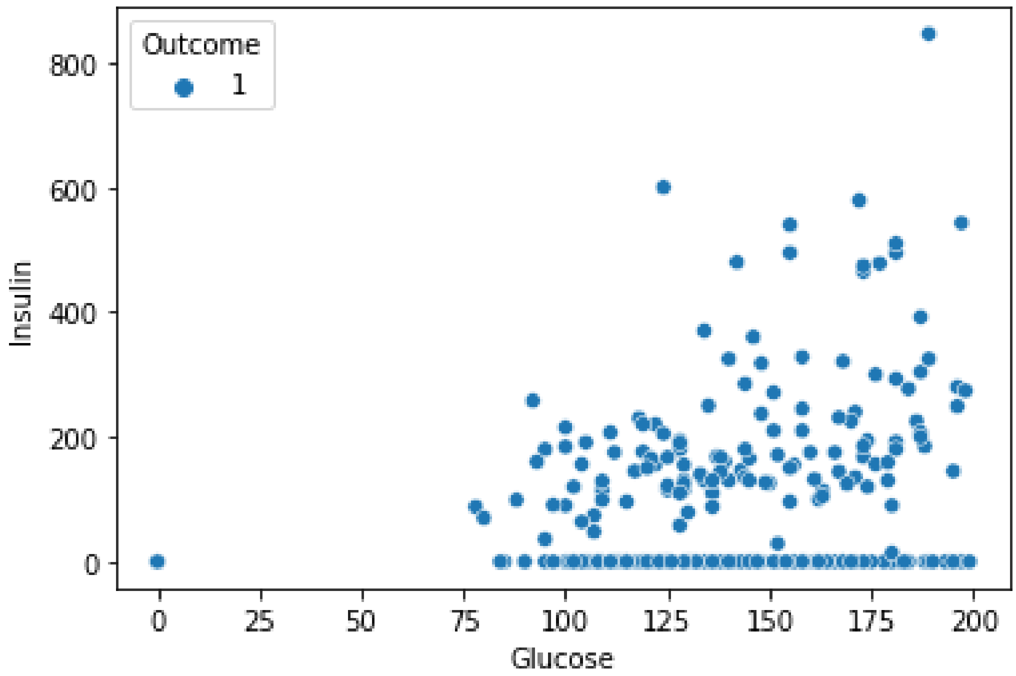
[23]:
```
sns.scatterplot(x = 'BMI', y = 'Glucose', data = diabetic_patients, hue =␣
↪'Outcome')
```

[23]: <AxesSubplot:xlabel='BMI', ylabel='Glucose'>




[24]:
```
sns.scatterplot(x = 'Glucose', y = 'Insulin', data = diabetic_patients, hue =␣
↪'Outcome')
```

[24]: <AxesSubplot:xlabel='Glucose', ylabel='Insulin'>

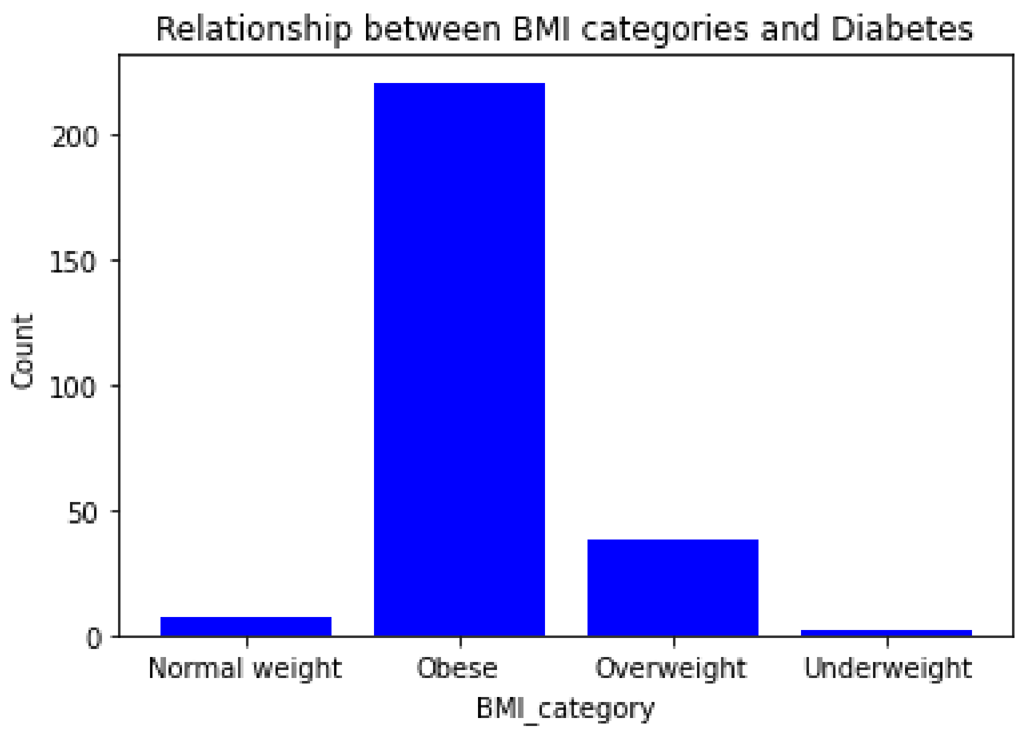



[25]:
```python
diabetic_patients.groupby('BMI_categories')['Outcome'].count()
```

[25]:
```
BMI_categories
Normal weight      7
Obese            221
Overweight        38
Underweight        2
Name: Outcome, dtype: int64
```

[26]:
```python
plt.bar(BMI_category_list, diabetic_patients.
 ↪groupby('BMI_categories')['Outcome'].count(), color = 'b')
plt.xlabel('BMI_category')
plt.ylabel('Count')
plt.title('Relationship between BMI categories and Diabetes')
```

[26]:
```
Text(0.5, 1.0, 'Relationship between BMI categories and Diabetes')
```




***Of the 268 diabetic individuals, 221 are Obese and 38 are Overweight, further explaining the risk of diabetes associated high BMI.***

### 0.1.2 Correlation Visualization

[27]:
```
data = df.copy()

data.corr()
```

[27]:

| | Pregnancies | Glucose | BloodPressure | SkinThickness |
|---|---|---|---|---|
| Pregnancies | 1.000000 | 0.129459 | 0.141282 | -0.081672 |
| Glucose | 0.129459 | 1.000000 | 0.152590 | 0.057328 |
| BloodPressure | 0.141282 | 0.152590 | 1.000000 | 0.207371 |
| SkinThickness | -0.081672 | 0.057328 | 0.207371 | 1.000000 |
| Insulin | -0.073535 | 0.331357 | 0.088933 | 0.436783 |
| BMI | 0.017683 | 0.221071 | 0.281805 | 0.392573 |
| DiabetesPedigreeFunction | -0.033523 | 0.137337 | 0.041265 | 0.183928 |
| Age | 0.544341 | 0.263514 | 0.239528 | -0.113970 |
| Outcome | 0.221898 | 0.466581 | 0.065068 | 0.074752 |

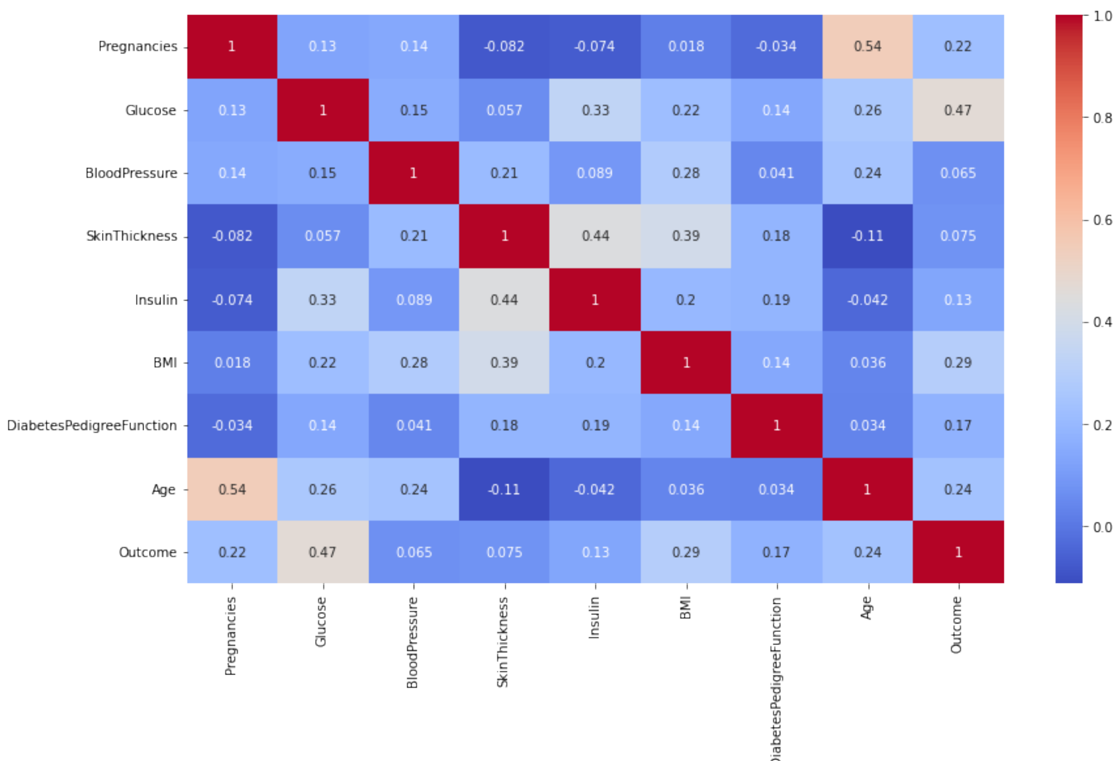
| | Insulin | BMI | DiabetesPedigreeFunction |
|---|---|---|---|
| Pregnancies | -0.073535 | 0.017683 | -0.033523 |
| Glucose | 0.331357 | 0.221071 | 0.137337 |
| BloodPressure | 0.088933 | 0.281805 | 0.041265 |

```
SkinThickness              0.436783  0.392573                  0.183928
Insulin                    1.000000  0.197859                  0.185071
BMI                        0.197859  1.000000                  0.140647
DiabetesPedigreeFunction   0.185071  0.140647                  1.000000
Age                       -0.042163  0.036242                  0.033561
Outcome                    0.130548  0.292695                  0.173844

                                Age   Outcome
Pregnancies                0.544341  0.221898
Glucose                    0.263514  0.466581
BloodPressure              0.239528  0.065068
SkinThickness             -0.113970  0.074752
Insulin                   -0.042163  0.130548
BMI                        0.036242  0.292695
DiabetesPedigreeFunction   0.033561  0.173844
Age                        1.000000  0.238356
Outcome                    0.238356  1.000000
```

[28]:
```
plt.figure(figsize = (14,8))
sns.heatmap(data.corr(), annot =True, cmap = 'coolwarm')
```

[28]: <AxesSubplot:>



***From this we can see that Glucose is the most correlated to the 'Outcome'***

## 0.2 Model Building

**Before building the model, further preparation will be done on the dataset**

[29]:
```
for col in data.columns:
    print(col)
    print(data[col].unique())
    print(data[col].nunique())
    print()
```

```
Pregnancies
[ 6  1  8  0  5  3 10  2  4  7  9 11 13 15 17 12 14]
17

Glucose
[148  85 183  89 137 116  78 115 197 125 110 168 139 189 166 100 118 107
 103 126  99 196 119 143 147  97 145 117 109 158  88  92 122 138 102  90
 111 180 133 106 171 159 146  71 105 101 176 150  73 187  84  44 141 114
  95 129  79   0  62 131 112 113  74  83 136  80 123  81 134 142 144  93
 163 151  96 155  76 160 124 162 132 120 173 170 128 108 154  57 156 153
 188 152 104  87  75 179 130 194 181 135 184 140 177 164  91 165  86 193
 191 161 167  77 182 157 178  61  98 127  82  72 172  94 175 195  68 186
 198 121  67 174 199  56 169 149  65 190]
136

BloodPressure
[ 72  66  64  40  74  50   0  70  96  92  80  60  84  30  88  90  94  76
  82  75  58  78  68 110  56  62  85  86  48  44  65 108  55 122  54  52
  98 104  95  46 102 100  61  24  38 106 114]
47

SkinThickness
[35 29  0 23 32 45 19 47 38 30 41 33 26 15 36 11 31 37 42 25 18 24 39 27
 21 34 10 60 13 20 22 28 54 40 51 56 14 17 50 44 12 46 16  7 52 43 48  8
 49 63 99]
51

Insulin
[  0  94 168  88 543 846 175 230  83  96 235 146 115 140 110 245  54 192
 207  70 240  82  36  23 300 342 304 142 128  38 100  90 270  71 125 176
  48  64 228  76 220  40 152  18 135 495  37  51  99 145 225  49  50  92
 325  63 284 119 204 155 485  53 114 105 285 156  78 130  55  58 160 210
 318  44 190 280  87 271 129 120 478  56  32 744 370  45 194 680 402 258
 375 150  67  57 116 278 122 545  75  74 182 360 215 184  42 132 148 180
 205  85 231  29  68  52 255 171  73 108  43 167 249 293  66 465  89 158
  84  72  59  81 196 415 275 165 579 310  61 474 170 277  60  14  95 237
```

```
 191 328 250 480 265 193  79  86 326 188 106  65 166 274  77 126 330 600
 185  25  41 272 321 144  15 183  91  46 440 159 540 200 335 387  22 291
 392 178 127 510  16 112]
186

BMI
[33.6 26.6 23.3 28.1 43.1 25.6 31.  35.3 30.5  0.  37.6 38.  27.1 30.1
 25.8 30.  45.8 29.6 43.3 34.6 39.3 35.4 39.8 29.  36.6 31.1 39.4 23.2
 22.2 34.1 36.  31.6 24.8 19.9 27.6 24.  33.2 32.9 38.2 37.1 34.  40.2
 22.7 45.4 27.4 42.  29.7 28.  39.1 19.4 24.2 24.4 33.7 34.7 23.  37.7
 46.8 40.5 41.5 25.  25.4 32.8 32.5 42.7 19.6 28.9 28.6 43.4 35.1 32.
 24.7 32.6 43.2 22.4 29.3 24.6 48.8 32.4 38.5 26.5 19.1 46.7 23.8 33.9
 20.4 28.7 49.7 39.  26.1 22.5 39.6 29.5 34.3 37.4 33.3 31.2 28.2 53.2
 34.2 26.8 55.  42.9 34.5 27.9 38.3 21.1 33.8 30.8 36.9 39.5 27.3 21.9
 40.6 47.9 50.  25.2 40.9 37.2 44.2 29.9 31.9 28.4 43.5 32.7 67.1 45.
 34.9 27.7 35.9 22.6 33.1 30.4 52.3 24.3 22.9 34.8 30.9 40.1 23.9 37.5
 35.5 42.8 42.6 41.8 35.8 37.8 28.8 23.6 35.7 36.7 45.2 44.  46.2 35.
 43.6 44.1 18.4 29.2 25.9 32.1 36.3 40.  25.1 27.5 45.6 27.8 24.9 25.3
 37.9 27.  26.  38.7 20.8 36.1 30.7 32.3 52.9 21.  39.7 25.5 26.2 19.3
 38.1 23.5 45.5 23.1 39.9 36.8 21.8 41.  42.2 34.4 27.2 36.5 29.8 39.2
 38.4 36.2 48.3 20.  22.3 45.7 23.7 22.1 42.1 42.4 18.2 26.4 45.3 37.
 24.5 32.2 59.4 21.2 26.7 30.2 46.1 41.3 38.8 35.2 42.3 40.7 46.5 33.5
 37.3 30.3 26.3 21.7 36.4 28.5 26.9 38.6 31.3 19.5 20.1 40.8 23.4 28.3
 38.9 57.3 35.6 49.6 44.6 24.1 44.5 41.2 49.3 46.3]
248

DiabetesPedigreeFunction
[0.627 0.351 0.672 0.167 2.288 0.201 0.248 0.134 0.158 0.232 0.191 0.537
 1.441 0.398 0.587 0.484 0.551 0.254 0.183 0.529 0.704 0.388 0.451 0.263
 0.205 0.257 0.487 0.245 0.337 0.546 0.851 0.267 0.188 0.512 0.966 0.42
 0.665 0.503 1.39  0.271 0.696 0.235 0.721 0.294 1.893 0.564 0.586 0.344
 0.305 0.491 0.526 0.342 0.467 0.718 0.962 1.781 0.173 0.304 0.27  0.699
 0.258 0.203 0.855 0.845 0.334 0.189 0.867 0.411 0.583 0.231 0.396 0.14
 0.391 0.37  0.307 0.102 0.767 0.237 0.227 0.698 0.178 0.324 0.153 0.165
 0.443 0.261 0.277 0.761 0.255 0.13  0.323 0.356 0.325 1.222 0.179 0.262
 0.283 0.93  0.801 0.207 0.287 0.336 0.247 0.199 0.543 0.192 0.588 0.539
 0.22  0.654 0.223 0.759 0.26  0.404 0.186 0.278 0.496 0.452 0.403 0.741
 0.361 1.114 0.457 0.647 0.088 0.597 0.532 0.703 0.159 0.268 0.286 0.318
 0.272 0.572 0.096 1.4   0.218 0.085 0.399 0.432 1.189 0.687 0.137 0.637
 0.833 0.229 0.817 0.204 0.368 0.743 0.722 0.256 0.709 0.471 0.495 0.18
 0.542 0.773 0.678 0.719 0.382 0.319 0.19  0.956 0.084 0.725 0.299 0.244
 0.745 0.615 1.321 0.64  0.142 0.374 0.383 0.578 0.136 0.395 0.187 0.905
 0.15  0.874 0.236 0.787 0.407 0.605 0.151 0.289 0.355 0.29  0.375 0.164
 0.431 0.742 0.514 0.464 1.224 1.072 0.805 0.209 0.666 0.101 0.198 0.652
 2.329 0.089 0.645 0.238 0.394 0.293 0.479 0.686 0.831 0.582 0.446 0.402
 1.318 0.329 1.213 0.427 0.282 0.143 0.38  0.284 0.249 0.926 0.557 0.092
 0.655 1.353 0.612 0.2   0.226 0.997 0.933 1.101 0.078 0.24  1.136 0.128
 0.422 0.251 0.677 0.296 0.454 0.744 0.881 0.28  0.259 0.619 0.808 0.34
```

```
 0.434 0.757 0.613 0.692 0.52  0.412 0.84  0.839 0.156 0.215 0.326 1.391
 0.875 0.313 0.433 0.626 1.127 0.315 0.345 0.129 0.527 0.197 0.731 0.148
 0.123 0.127 0.122 1.476 0.166 0.932 0.343 0.893 0.331 0.472 0.673 0.389
 0.485 0.349 0.279 0.346 0.252 0.243 0.58  0.559 0.302 0.569 0.378 0.385
 0.499 0.306 0.234 2.137 1.731 0.545 0.225 0.816 0.528 0.509 1.021 0.821
 0.947 1.268 0.221 0.66  0.239 0.949 0.444 0.463 0.803 1.6   0.944 0.196
 0.241 0.161 0.135 0.376 1.191 0.702 0.674 1.076 0.534 1.095 0.554 0.624
 0.219 0.507 0.561 0.421 0.516 0.264 0.328 0.233 0.108 1.138 0.147 0.727
 0.435 0.497 0.23  0.955 2.42  0.658 0.33  0.51  0.285 0.415 0.381 0.832
 0.498 0.212 0.364 1.001 0.46  0.733 0.416 0.705 1.022 0.269 0.6   0.571
 0.607 0.17  0.21  0.126 0.711 0.466 0.162 0.419 0.63  0.365 0.536 1.159
 0.629 0.292 0.145 1.144 0.174 0.547 0.163 0.738 0.314 0.968 0.409 0.297
 0.525 0.154 0.771 0.107 0.493 0.717 0.917 0.501 1.251 0.735 0.804 0.661
 0.549 0.825 0.423 1.034 0.16  0.341 0.68  0.591 0.3   0.121 0.502 0.401
 0.601 0.748 0.338 0.43  0.892 0.813 0.693 0.575 0.371 0.206 0.417 1.154
 0.925 0.175 1.699 0.682 0.194 0.4   0.1   1.258 0.482 0.138 0.593 0.878
 0.157 1.282 0.141 0.246 1.698 1.461 0.347 0.362 0.393 0.144 0.732 0.115
 0.465 0.649 0.871 0.149 0.695 0.303 0.61  0.73  0.447 0.455 0.133 0.155
 1.162 1.292 0.182 1.394 0.217 0.631 0.88  0.614 0.332 0.366 0.181 0.828
 0.335 0.856 0.886 0.439 0.253 0.598 0.904 0.483 0.565 0.118 0.177 0.176
 0.295 0.441 0.352 0.826 0.97  0.595 0.317 0.265 0.646 0.426 0.56  0.515
 0.453 0.785 0.734 1.174 0.488 0.358 1.096 0.408 1.182 0.222 1.057 0.766
 0.171]
517

Age
[50 31 32 21 33 30 26 29 53 54 34 57 59 51 27 41 43 22 38 60 28 45 35 46
 56 37 48 40 25 24 58 42 44 39 36 23 61 69 62 55 65 47 52 66 49 63 67 72
 81 64 70 68]
52

Outcome
[1 0]
2

BMI_categories
['Obese' 'Overweight' 'Normal weight' 'Underweight']
4
```

From the output above we can see that some of the columns contain the value 0, which may be an error.

For example BloodPressure of 0 may mean that a person is dead. Likewise Glucose level, SkinThickness and so on

[30]:
```
# Replace 0's with NAN
columns = ['Glucose', 'BloodPressure', 'SkinThickness', 'Insulin', 'BMI']
```

```
for column in columns:
    data[column] = data[column].replace(0,np.NAN)
```

[31]:
```
data
```

[31]:
```
     Pregnancies  Glucose  BloodPressure  SkinThickness  Insulin   BMI  \
0              6    148.0           72.0           35.0      NaN  33.6
1              1     85.0           66.0           29.0      NaN  26.6
2              8    183.0           64.0            NaN      NaN  23.3
3              1     89.0           66.0           23.0     94.0  28.1
4              0    137.0           40.0           35.0    168.0  43.1
..           …       …             …              …        …    …
763           10    101.0           76.0           48.0    180.0  32.9
764            2    122.0           70.0           27.0      NaN  36.8
765            5    121.0           72.0           23.0    112.0  26.2
766            1    126.0           60.0            NaN      NaN  30.1
767            1     93.0           70.0           31.0      NaN  30.4

     DiabetesPedigreeFunction  Age  Outcome BMI_categories
0                       0.627   50        1          Obese
1                       0.351   31        0     Overweight
2                       0.672   32        1  Normal weight
3                       0.167   21        0     Overweight
4                       2.288   33        1          Obese
..                        …  …      …            …
763                     0.171   63        0          Obese
764                     0.340   27        0          Obese
765                     0.245   30        0     Overweight
766                     0.349   47        1          Obese
767                     0.315   23        0          Obese

[768 rows x 10 columns]
```

[32]:
```
data.isnull().sum()
```

[32]:
```
Pregnancies                   0
Glucose                       5
BloodPressure                35
SkinThickness               227
Insulin                     374
BMI                          11
DiabetesPedigreeFunction      0
Age                           0
Outcome                       0
BMI_categories                0
dtype: int64
```

[33]:
```
# Replacing null elements with the mean
for column in columns:
    data[column].fillna(data[column].mean(), inplace = True)
```

[34]:
```
data.isnull().sum()
```

[34]:
```
Pregnancies                 0
Glucose                     0
BloodPressure               0
SkinThickness               0
Insulin                     0
BMI                         0
DiabetesPedigreeFunction    0
Age                         0
Outcome                     0
BMI_categories              0
dtype: int64
```

### 0.2.1 TRAIN - TEST SPLIT

[35]:
```
x = data.drop(columns = ['Outcome', 'BMI_categories'])
y = data['Outcome']
```

[36]:
```
x.head()
```

[36]:
```
   Pregnancies  Glucose  BloodPressure  SkinThickness     Insulin   BMI  \
0            6    148.0           72.0       35.00000  155.548223  33.6
1            1     85.0           66.0       29.00000  155.548223  26.6
2            8    183.0           64.0       29.15342  155.548223  23.3
3            1     89.0           66.0       23.00000   94.000000  28.1
4            0    137.0           40.0       35.00000  168.000000  43.1

   DiabetesPedigreeFunction  Age
0                     0.627   50
1                     0.351   31
2                     0.672   32
3                     0.167   21
4                     2.288   33
```

[37]:
```
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size = 0.2,␣
↪random_state = 42)
```

[38]:
```
len(x_train), len(y_train), len(x_test), len(y_test)
```

[38]:
```
(614, 614, 154, 154)
```

### 0.2.2 Logistic Regression Model

[39]:
```
model = LogisticRegression()
model.fit(x_train, y_train)
```

[39]:
```
LogisticRegression()
```

[40]:
```
prediction = model.predict(x_test)
```

[41]:
```
prediction
```

[41]:
```
array([0, 0, 0, 0, 0, 1, 0, 0, 1, 1, 0, 1, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0,
       0, 0, 1, 1, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 0, 0, 1, 0, 1, 1, 0,
       0, 1, 1, 0, 0, 1, 0, 1, 1, 0, 0, 0, 1, 0, 0, 1, 1, 0, 0, 0, 0, 1,
       0, 1, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 1, 0,
       0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 1, 0, 1, 0, 1, 1, 1, 0, 0, 1, 0, 0,
       0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1,
       0, 0, 1, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0],
      dtype=int64)
```

### 0.2.3 KNeighborsClassifier Model

[42]:
```
k_model = KNeighborsClassifier(n_neighbors=7)
k_model.fit(x_train, y_train)
```

[42]:
```
KNeighborsClassifier(n_neighbors=7)
```

[43]:
```
k_prediction = k_model.predict(x_test)
k_prediction
```

[43]:
```
array([0, 1, 0, 0, 0, 1, 0, 1, 1, 0, 1, 1, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0,
       1, 0, 1, 1, 0, 0, 0, 0, 1, 0, 1, 1, 0, 1, 1, 0, 0, 0, 0, 1, 1, 1,
       0, 0, 0, 0, 0, 1, 0, 1, 1, 1, 0, 0, 1, 0, 0, 1, 1, 0, 0, 0, 0, 1,
       0, 1, 0, 1, 1, 0, 0, 0, 0, 1, 1, 0, 0, 0, 1, 0, 0, 1, 0, 1, 1, 0,
       0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 1, 0, 0, 0, 0, 1, 1, 0, 1, 1, 0, 1,
       0, 1, 0, 1, 1, 0, 1, 0, 0, 1, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 0,
       0, 1, 1, 1, 0, 1, 1, 0, 0, 0, 1, 1, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0],
      dtype=int64)
```

### 0.2.4 GradientBoostingClassifier

[44]:
```
GB_model = GradientBoostingClassifier()
GB_model.fit(x_train, y_train)
```

[44]:
```
GradientBoostingClassifier()
```

[45]:
```
GB_prediction = GB_model.predict(x_test)
GB_prediction
```

[45]:

```
array([1, 0, 0, 0, 0, 1, 0, 1, 1, 1, 0, 1, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0,
       1, 0, 1, 1, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 1, 0,
       0, 1, 1, 0, 0, 1, 0, 1, 1, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 1, 0, 1,
       0, 1, 0, 1, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 1, 0,
       0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 1, 0, 1, 0, 1, 1, 1, 0, 0, 1, 0, 1,
       0, 1, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1,
       0, 0, 1, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0],
      dtype=int64)
```

### 0.2.5 Measuring Performance of the Models

**For Logistic Regression Model**

[46]:

```
metrics = {
    'Accuracy': accuracy_score,
    'Precision': precision_score,
    'Recall': recall_score,
    'F1 score': f1_score,
    'Confusion Matrix': confusion_matrix
    }

for metric_name, metric_func in metrics.items():
    if metric_name == 'Confusion Matrix':
        print(metric_name)
        print(metric_func(y_test, prediction))
        print('\n')
    elif metric_name == 'Accuracy':
        print(metric_name)
        print(metric_func(y_test, prediction))
        print('\n')
    elif metric_name == 'Precision':
        print(metric_name)
        print(metric_func(y_test, prediction))
        print('\n')
    elif metric_name == 'Recall':
        print(metric_name)
        print(metric_func(y_test, prediction))
        print('\n')
    else:
        print(metric_name)
        print(metric_func(y_test, prediction))
        print('\n')
```

```
Accuracy
0.7532467532467533



Precision
0.660377358490566
```

```
Recall
0.6363636363636364


F1 score
0.6481481481481481


Confusion Matrix
[[81 18]
 [20 35]]
```

**For KNeighborsClassifier**

[47]:
```python
for metric_name, metric_func in metrics.items():
    if metric_name == 'Confusion Matrix':
        print(metric_name)
        print(metric_func(y_test, k_prediction))
        print('\n')
    elif metric_name == 'Accuracy':
        print(metric_name)
        print(metric_func(y_test, k_prediction))
        print('\n')
    elif metric_name == 'Precision':
        print(metric_name)
        print(metric_func(y_test, k_prediction))
        print('\n')
    elif metric_name == 'Recall':
        print(metric_name)
        print(metric_func(y_test, k_prediction))
        print('\n')
    else:
        print(metric_name)
        print(metric_func(y_test, k_prediction))
        print('\n')
```

```
Accuracy
0.6558441558441559


Precision
0.5151515151515151


Recall
```

```
0.6181818181818182


F1 score
0.5619834710743802


Confusion Matrix
[[67 32]
 [21 34]]
```

**For GradientBoostingClassifier**

[48]:
```python
for metric_name, metric_func in metrics.items():
    if metric_name == 'Confusion Matrix':
        print(metric_name)
        print(metric_func(y_test, GB_prediction))
        print('\n')
    elif metric_name == 'Accuracy':
        print(metric_name)
        print(metric_func(y_test, GB_prediction))
        print('\n')
    elif metric_name == 'Precision':
        print(metric_name)
        print(metric_func(y_test, GB_prediction))
        print('\n')
    elif metric_name == 'Recall':
        print(metric_name)
        print(metric_func(y_test, GB_prediction))
        print('\n')
    else:
        print(metric_name)
        print(metric_func(y_test, GB_prediction))
        print('\n')
```

```
Accuracy
0.7337662337662337


Precision
0.6129032258064516


Recall
0.6909090909090909
```

```
F1 score
0.6495726495726496


Confusion Matrix
[[75 24]
 [17 38]]
```

```
[ ]:
```